3.5 Scalar Types

Scalar types comprise enumeration types, integer types and real types.
For every scalar subtype S, the following attributes are defined:

S'First denotes the lower bound of the range of S.
The value of this attribute is of the type of S.

S'Last denotes the upper bound of the range of S.
The value of this attribute is of the type of S.

S'Range is equivalent to the range S'First .. S'Last.

S'Base denotes an unconstrained subtype of the type of S.
This unconstrained subtype is called the base subtype of the type.

S'Min denotes a function with the following specification:

```
function S'Min(Left, Right : S'Base) return S'Base
```

The function returns the lesser of the values of the two parameters.

S'Max denotes a function with the following specification:

```
function S'Max(Left, Right : S'Base) return S'Base
```

The function returns the greater of the values of the two parameters.

S'Succ denotes a function with the following specification:

```
function S'Succ(Arg : S'Base) return S'Base
```

For an enumeration type, the function returns the value whose position number is one more than that of the value of Arg; Constraint_Error is raised if there is no such value of the type.

For an integer type, the function returns the result of adding one to the value of Arg.

For a fixed point type, the function returns the result of adding small to the value of Arg.

For a floating point type, the function returns the machine number (as defined in 3.5.7) immediately above the value of Arg; Constraint_Error is raised if there is no such machine number.

S'Pred denotes a function with the following specification:

```
function S'Pred(Arg : S'Base) return S'Base
```

For an enumeration type, the function returns the value whose position number is one less than that of the value of Arg; Constraint_Error is raised if there is no such value of the type.

For an integer type, the function returns the result of subtracting one from the value of Arg.

For a fixed point type, the function returns the result of subtracting small from the value of Arg.

For a floating point type, the function returns the machine number (as defined in 3.5.7) immediately below the value of Arg; Constraint_Error is raised if there is no such machine number.

S'Wide_Image denotes a function with the following specification:

```
function S'Wide_Image(Arg : S'Base)
return Wide_String
```

The function returns an image of the value of Arg, that is, a sequence of characters representing the value in display form. The lower bound of the result is one.

The image of an integer value is the corresponding decimal literal, without underlines, leading zeros, exponent, or trailing spaces, but with a single leading character that is either a minus sign or a space.

The image of an enumeration value is either the corresponding identifier in upper case or the corresponding character literal (including the two apostrophes); neither leading nor trailing spaces are included.

For a nongraphic character (a value of a character type that has no enumeration literal associated with it), the result is a corresponding language-defined or implementation-defined name in upper case (for example, the image of the nongraphic character identified as nul is ‘‘NUL’’ - the quotes are not part of the image).

The image of a floating point value is a decimal real literal best approximating the value (rounded away from zero if halfway between) with a single leading character that is either a minus sign or a space, a single digit (that is nonzero unless the value is zero), a decimal point, S'Digits-1 (see 3.5.8) digits after the decimal point (but one if S'Digits is one), an upper case E, the sign of the exponent (either + or -), and two or more digits (with leading zeros if necessary) representing the exponent. If S'Signed_Zeros is True, then the leading character is a minus sign for a negatively signed zero.

The image of a fixed point value is a decimal real literal best approximating the value (rounded away from zero if halfway between) with a single leading character that is either a minus sign or a space, one or more digits before the decimal point (with no redundant leading zeros), a decimal point, and S'Aft (see 3.5.10) digits after the decimal point.

S'Image denotes a function with the following specification:

```
function S'Image(Arg : S'Base) return String
```

The function returns an image of the value of Arg as a String.

The lower bound of the result is one. The image has the same sequence of graphic characters as that defined for S'Wide_Image if all the graphic characters are defined in Character; otherwise the sequence of characters is implementation defined (but no shorter than that of S'Wide_Image for the same value of Arg).

S'Wide_Width denotes the maximum length of a Wide_String returned by S'Wide_Image over all values of the subtype S. It denotes zero for a subtype that has a null range. Its type is universal_integer.

S'Width denotes the maximum length of a String returned by S'Image over all values of the subtype S. It denotes zero for a subtype that has a null range. Its type is universal_integer.

S'Wide_Value denotes a function with the following specification:

```
function S'Wide_Value(Arg : Wide_String)
return S'Base
```

This function returns a value given an image of the value as a Wide_String, ignoring any leading or trailing spaces.

For the evaluation of a call on S'Wide_Value for an enumeration subtype S, if the sequence of characters of the parameter (ignoring leading and trailing spaces) has the syntax of an enumeration literal and if it corresponds to a literal of the type of S (or corresponds to the result of S'Wide_Image for a nongraphic character of the type), the result is the corresponding enumeration value; otherwise Constraint_Error is raised.

For the evaluation of a call on S'Wide_Value (or S'Value) for an integer subtype S, if the sequence of characters of the parameter (ignoring leading and trailing spaces) has the syntax of an integer literal, with an optional leading sign character (plus or minus for a signed type; only plus for a modular type), and the corresponding numeric value belongs to the base range of the type of S, then that value is the result; otherwise Constraint_Error is raised.

For the evaluation of a call on S'Wide_Value (or S'Value) for a real subtype S, if the sequence of characters of the parameter (ignoring leading and trailing spaces) has the syntax of one of the following:

- numeric_literal
- numeral.[exponent]
- numeral[exponent]
- base#based_numeral.#[exponent]
- base#.based_numeral#[exponent]

with an optional leading sign character (plus or minus), and if the corresponding numeric value belongs to the base range of the type of S, then that value is the result; otherwise Constraint_Error is raised. The sign of a zero value is preserved (positive if none has been specified) if S'Signed_Zeros is True.

S'Value denotes a function with the following specification:

```
function S'Value(Arg : String) return S'Base
```

This function returns a value given an image of the value as a String, ignoring any leading or trailing spaces.

For the evaluation of a call on S'Value for an enumeration subtype S, if the sequence of characters of the parameter (ignoring leading and trailing spaces) has the syntax of an enumeration literal and if it corresponds to a literal of the type of S (or corresponds to the result of S'Image for a value of the type), the result is the corresponding enumeration value; otherwise Constraint_Error is raised.

For a numeric subtype S, the evaluation of a call on S'Value with Arg of type String is equivalent to a call on S'Wide_Value for a corresponding Arg of type Wide_String.

3.5.5 Operations of Discrete Types

Discrete types comprise enumeration types and integer types.
For every discrete subtype S, the following attributes are defined:

S'Pos denotes a function with the following specification:

```
function S'Pos(Arg : S'Base)
  return universal_integer
```

This function returns the position number of the value of Arg, as a value of type universal_integer.

S'Val denotes a function with the following specification:

```
function S'Val(Arg : universal_integer)
  return S'Base
```

This function returns a value of the type of S whose position number equals the value of Arg. For the evaluation of a call on S'Val, if there is no value in the base range of its type with the given position number, Constraint_Error is raised.

NOTES:

- Indexing and loop iteration use values of discrete types.
- The predefined operations of a discrete type include the assignment operation, qualification, the membership tests, and the relational operators; for a boolean type they include the short-circuit control forms and the logical operators; for an integer type they include type conversion to and from other numeric types, as well as the binary and unary adding operators - and +, the multiplying operators, the unary operator abs, and the exponentiation operator
- The following relations are satisfied (in the absence of an exception) by these attributes:

```
S'Val(S'Pos(X)) = X
S'Pos(S'Val(N)) = N
```

EXAMPLES

Examples of enumeration types and subtypes:

```
type Day    is (Mon, Tue, Wed, Thu, Fri, Sat, Sun);
type Suit   is (Clubs, Diamonds, Hearts, Spades);
type Gender is (M, F);
type Level  is (Low, Medium, Urgent);
type Color  is (White, Red, Yellow, Green, Blue, Brown, Black);
type Light  is (Red, Amber, Green); -- Red and Green are overloaded

type Hexa   is ('A', 'B', 'C', 'D', 'E', 'F');
type Mixed  is ('A', 'B', '*', B, None, '?', '%');

subtype Weekday is Day   range Mon .. Fri;
subtype Major   is Suit  range Hearts .. Spades;
subtype Rainbow is Color range Red .. Blue; -- the Color Red, not Light
```

Examples of attributes of discrete subtypes:

```
-- For the types and subtypes declared above the following hold:

-- Color'First   = White,     Color'Last   = Black
-- Rainbow'First = Red,       Rainbow'Last = Blue

-- Color'Succ(Blue) = Rainbow'Succ(Blue) = Brown
-- Color'Pos(Blue)  = Rainbow'Pos(Blue)  = 4
-- Color'Val(0)     = Rainbow'Val(0)     = White
```

3.5.8 Operations of Floating Point Types

The following attribute is defined for every floating point subtype S:

S'Digits denotes the requested decimal precision for the subtype S.
The value of this attribute is of the type universal_integer.

The requested decimal precision of the base subtype of a floating point type T is defined to be the largest value of d for which

ceiling(d * log(10) / log(T'Machine_Radix)) + {g}[1] <= T'Model_Mantissa

where g is 0 if Machine_Radix is a positive power of 10 and 1 otherwise.

3.5.10 Operations of Fixed Point Types

The following attributes are defined for every fixed point subtype S:

S'Small denotes the small of the type of S.
The value of this attribute is of the type universal_real.

S'Delta denotes the delta of the fixed point subtype S.
The value of this attribute is of the type universal_real.

S'Fore yields the minimum number of characters needed before the decimal point for the decimal representation of any value of the subtype S, assuming that the representation does not include an exponent, but includes a one-character prefix that is either a minus sign or a space. (This minimum number does not include superfluous zeros or underlines, and is at least 2.) The value of this attribute is of the type universal_integer.

S'Aft yields the number of decimal digits needed after the decimal point to accommodate the delta of the subtype S, unless the delta of the subtype S is greater than 0.1, in which case the attribute yields the value one. (S'Aft is the smallest positive integer N for which (10**N)*S'Delta is greater than or equal to one.) The value of this attribute is of the type universal_integer.

The following additional attributes are defined for every decimal fixed point subtype S:

S'Digits denotes the digits of the decimal fixed point subtype S, which corresponds to the number of decimal digits that are representable in objects of the subtype. The value of this attribute is of the type universal_integer. Its value is determined as follows:

- For a first subtype or a subtype defined by a subtype_indication with a digits_constraint, the digits is the value of the expression given after the reserved word digits;
- For a subtype defined by a subtype_indication without a digits_constraint, the digits of the subtype is the same as that of the subtype denoted by the subtype_mark in the subtype_indication.
- The digits of a base subtype is the largest integer D such that the range -(10**D-1)*delta .. +(10**D-1)*delta is included in the base range of the type.

S'Scale denotes the scale of the subtype S, defined as the value N such that S'Delta = 10.0**(-N). The scale indicates the position of the point relative to the rightmost significant digits of values of subtype S. The value of this attribute is of the type universal_integer.

S'Round denotes a function with the following specification:

```
function S'Round(X : universal_real) return S'Base
```

The function returns the value obtained by rounding X (away from 0, if X is midway between two values of the type of S).

3.6.2 Operations of Array Types

NOTE: The argument N used in the attribute_designators for the N-th dimension of an array shall be a static expression of some integer type. The value of N shall be positive (nonzero) and no greater than the dimensionality of the array.

A'First denotes the lower bound of the first index range;
its type is the corresponding index type.

A'First(N) denotes the lower bound of the N-th index range;
its type is the corresponding index type.

A'Last denotes the upper bound of the first index range;
its type is the corresponding index type.

A'Last(N) denotes the upper bound of the N-th index range;
its type is the corresponding index type.

A'Range is equivalent to the range A'First .. A'Last,
except that the prefix A is only evaluated once.

A'Range(N) is equivalent to the range A'First(N) .. A'Last(N),
except that the prefix A is only evaluated once.

3.6.2 Operations of Array Types (continued)

NOTE: The argument N used in the attribute_designators for the N-th dimension of an array shall be a static expression of some integer type. The value of N shall be positive (nonzero) and no greater than the dimensionality of the array.

A'Length denotes the number of values of the first index range (zero for a null range); its type is universal_integer.

A'Length(N) denotes the number of values of the N-th index range (zero for a null range); its type is universal_integer.

3.10.2 Operations of Access Types

The following attribute is defined for a prefix X that denotes an aliased view of an object:

X'Access yields an access value that designates the object denoted by X.
The type of X'Access is an access-to-object type, as determined by the expected type. The expected type shall be a general access type. X shall denote an aliased view of an object, including possibly the current instance (see 8.6) of a limited type within its definition, or a formal parameter or generic formal object of a tagged type.

The following attribute is defined for a prefix P that denotes a subprogram:

P'Access yields an access value that designates the subprogram denoted by P.
The type of P'Access is an access-to-subprogram type (S), as determined by the expected type. The accessibility level of P shall not be statically deeper than that of S. In addition to the places where Legality Rules normally apply (see 12.3), this rule applies also in the private part of an instance of a generic unit. The profile of P shall be subtype-conformant with the designated profile of S, and shall not be Intrinsic. If the subprogram denoted by P is declared within a generic body, S shall be declared within the generic body.

9.9 Task and Entry Attributes

For a prefix T that is of a task type (after any implicit dereference), the following attributes are defined:

T'Callable yields the value True when the task denoted by T is callable, and False otherwise; a task is callable unless it is completed or abnormal. The value of this attribute is of the predefined type Boolean.

T'Terminated yields the value True if the task denoted by T is terminated, and False otherwise. The value of this attribute is of the predefined type Boolean.

For a prefix E that denotes an entry of a task or protected unit, the following attribute is defined. This attribute is only allowed within the body of the task or protected unit, but excluding, in the case of an entry of a task unit, within any program unit that is, itself, inner to the body of the task unit.

E'Count Yields the number of calls presently queued on the entry E of the current instance of the unit. The value of this attribute is of the type universal_integer.